

Google Summer of Code 2009 Proposal

Project: fast water and gas simulation

Brief explanation: Currently Step has molecular dynamics based simulation of gas and water (that is a gas is simulated as a collection of particles). This is very useful for demonstrating microscopical properties of the gas as well as its connection with macroscopical quantities like temperature. But this is far from optimal to demonstrate things like a boat floating in the water, water flowing from a glass, etc. This project involves:

- investigate fast methods of simulating water and gas: ranging from molecular dynamics but with simplified potentials, various optimizations, coarse graining methods, to lattice Boltzmann methods; investigate existing libraries for water simulation (for example take a look at elbeem library from Blender)
- implement selected method of simulation or incorporate selected library in StepCore
- implement GUI in Step for creating and modifying macroscopical quantities of gas and water

Expected results: Ability to easily simulate in Step experiments like a boat floating in the water, water flowing from a glass, etc.

Knowledge Pre-Requisite: Required: C++, physics, good knowledge of numerical methods. Could be useful: Qt.

Mentor: Vladimir Kuznetsov <ks dot vladimir at gmail dot com>

Name:
Christopher Ing

Email Address:
ing.chris@gmail.com

Freenode IRC Nick:
spctme

Location (City, Country and/or Time Zone):
Kitchener, Ontario
North America EST / GMT -5

Proposal Name:
Fast Water and Gas Simulation in KDE Step

Motivation for Proposal / Goal:

The goal of the KDE Education Project is to provide open-source tools to enhance learning for students of all ages. The Step application, within this software suite, provides object-oriented physics simulation based on the principles of classical mechanics. As demonstrated by the creative example experiments included with Step, the educational value of this program is high. However, when comparing Step to the capabilities of existing physics sandboxes (Phun, Box2D, etc.), there are many areas for improvement.

By implementing fast fluid simulation, an entirely new class of physics demonstrations will be available for students. In the following figures, I illustrate some of the real-time fluid simulations demonstrations we wish to reproduce. In **Fig. 1**, fluid particles drive the motion of a water wheel. It is our goal to have water interact with other rigid bodies in Step with a high degree of realism.

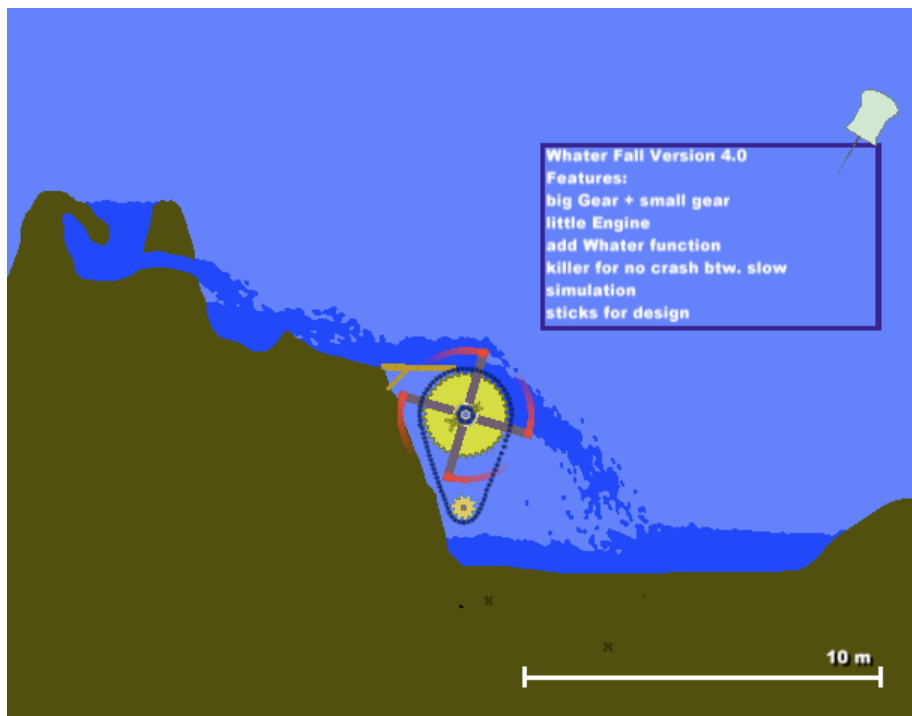


Fig.1 Phun User-Created Level with a realistic water wheel. The interaction of the fluid particles with rigid bodies will be a topic of careful research.

More fluid-body collision is further demonstrated in **Fig. 2**. Early on in this simulation we also see the solid body floating on the surface of the water by an effective buoyancy force. We aim to reproduce this physical phenomena and ideally have the user be able to adjust the density of fluid in real-time. As shown in **Fig 3**., we also intend have user adjustable settings for our fluid. In this demonstration we see the user adjust viscosity and bounciness of the fluid in real-time. More specifically, it would be ideal to have the user be able to select an area of a fluid and adjust the density at that point.

Water Simulation

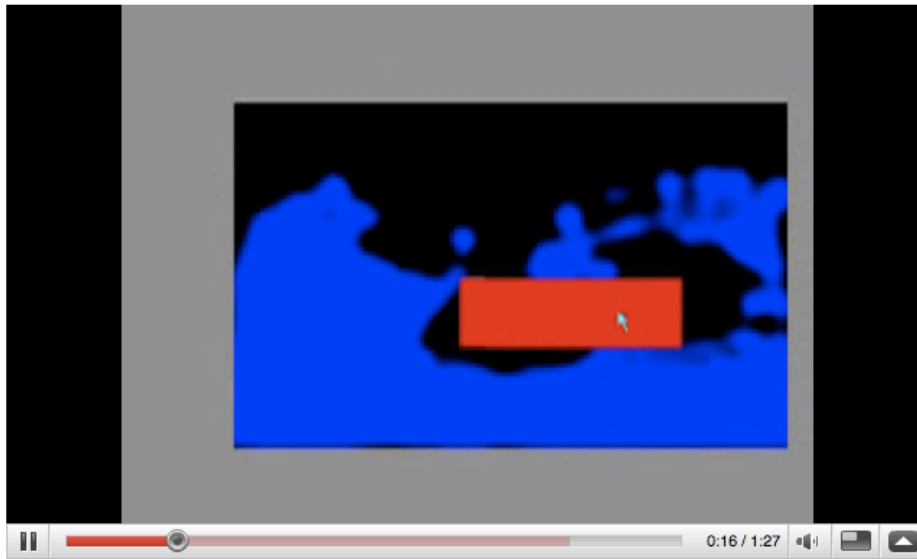


Fig.2 Fluid Simulation by Danny Chapman.
(<http://www.youtube.com/watch?v=T0E6e7viXyc>)

Flui°D°emo - A 2D Fluid Simulation



Fig 3. A fluid simulation with the adjustable Viscosity, “Bounciness”
(<http://www.youtube.com/watch?v=LpvTM1wiwlk>)

Along with traditional liquid simulation, the simulation of a gaseous fluid should be possible using many of the same principles. The current gas simulation in Step can be improved using the same method outlined for liquid, allowing for low-velocity low-density cloud-like masses. Realistic gas-like fluids should function exactly how the current Gas module performs but allow for a much larger number

of particles displayed on the screen by means of a more efficient method of calculating particle-particle interactions.

While most physics simulators were designed for purely entertainment purposes, Step is, currently, primarily an educational tool. Among many features, what I noticed first, that differentiates it from other physics simulators, is that it calculates both user-inputted and numerical based errors. Successful fluid implementation will maintain this level of realism but make using Step a more pleasurable and entertaining learning experience. This project will bring Step to a new level of cutting-edge computational physics which will make it desirable by educators world-wide.

Implementation Details:

Upon discovering this project, I have researched real-time fluid simulation in great detail, while keeping in correspondence with Vladimir Kuznetsov, my prospective mentor. As demonstrated in the previous examples, there have been a number of successes in real-time 2D computational fluid dynamics. However, this topic is an extremely complicated issue that is currently being researched by scientists in fields ranging from computer graphics, astrophysics, and oceanographers, to aviation engineers.

There are complicated grid-based methods, such as the Lattice-Boltzman method, for simulating incompressible flow, in order to reproduce viscosity and turbulence effects. However, in Step, we are interested in a more simplistic view of a fluid as a collection of individual particles. In all the previous examples, including the popular physics simulator Phun, the technique of Smoothed Particle Hydrodynamics (SPH) is used for a large number of particles without significant slow-down on modern computer architecture.

Gas simulation is currently implemented in Step. Each Gas object is comprised of a number of individual Gas Particles (which inherit the properties of the Particle object) initially confined to a rectangular region. As the World evolves in time, GasLJForces are calculated which are integrated to determine the new gas particle positions. Using the properties of statistical mechanics it is possible to calculate macroscopic quantities based on this ensemble like temperature. However, this object is best used to describe a “diffuse gas” and not a dense fluid.

Using the same design scheme, a particle-based fluid can be implemented. I intend to design a new StepCore class called Fluid which is comprised of Fluid Particles. The Particles will interact with a FluidForce which will have numerical error associated with it. Each Fluid Particle will possess the following quantities that will be used for our Smoothed Particle Hydrodynamics fluid calculations:

- Mass
- Position
- Velocity
- Pressure
- Density
- Force

Conveniently, all of these member functions with the exception of density and pressure already exist in the Particle class of StepCore so inheritance is a natural choice.

Although, the greatest challenge will likely be integrating the SPH fluid physics to the Fluid object. Smoothed Particle Hydrodynamics is a Lagrangian solution to the standard partial differential equations that characterize fluids[1]. The steps for this process are described in pseudocode [2][1] are:

```
For all particles
  -compute the density and pressure of all particles
  -compute the forces on all particles
  -advance the simulation by timestep dt by integration
```

This technique works by using a weighting function to consider neighboring particle densities, pressures and forces. With the exception of the density/pressure computation step, there is nothing new about the time evolution of a fluid compared to a gas. The World:doEvolve() function of StepCore shouldn't require any modification in order to propagate the fluid motion.

However, the focus of this coding project is not to recode this algorithm. To gain the maximum performance and realism, I intend to implement the GPL SPH library FLUIDS written by R. Hoetzlein (<http://www.rchoetzlein.com/eng/graphics/fluids.htm>). This 3D library has been optimized and benchmarked against other SPH models [1] with great performance. Additionally, if Step moves from a 2D to 3D environment we know this algorithm should function efficiently. Porting this library from 3D to 2D should not be extremely difficult since the library is well documented.

Another challenging aspect of this project will be optimizing the graphical representation of a fluid. Borrowing from the ideas depicted in physics simulators like Phun, I should be able to draw a surface around the outer edge of my Fluid instead of depicting individual particles. This could be achieved by placing an effective circle around each Fluid Particle and allow these spheres to overlap. Clicking on a specific sphere will allow the user to adjust that FluidParticles individual density and pressure.

Once the liquid fluid object is implemented, I need to assess the potential for this Fluid object to replace the current Gas module in Step. That is, by adjusting the parameters of the ensemble of FluidParticles can I reproduce the low-density motion of an ensemble of classical particles? If the liquid fluid is not suitable, Vladimir offered a number of suggestions. By introducing a cut-off distance or perhaps a toggle for the Lennard Jones particle-particle interaction, we will be able to optimize the Gas implementation for a larger number of particles.

Also, regardless of what goes on “behind-the-scenes”, using the same graphics pipeline that renders the surface around the Fluid object, we will be able to illustrate the gaseous ensemble more realistically as a cloud of overlapping circles.

In summary, with the Gas and GasParticle class as a template, I will implement the FLUIDS library ported from 3D to 2D in order to create the new Fluid and FluidParticle objects in StepCore. Additionally I will have to research error calculations for this simulation.

[1] Hoetzlein, R., Hollerer, T. Analysing Performance and Efficiency of Smoothed Particle Hydrodynamics. Graduate Workshop in Computer Science, UC Santa Barbara (2008).

[2] M. Roy, T. Physically-Based Fluid Modeling using Smoothed Particle Hydrodynamics. University of Illinois (1995)

Timeline

Stage 1 – Planning (Weeks 1-2)

- Write out initial documentation for Fluid, FluidParticle, FluidForce classes
 - This includes, member functions, member variables, inheritance
- Gain a greater understanding of the Fluids Library
 - Determine what core calculations are performed and where 3D objects are being used.
- Understanding how the Qt GUI is designed for Step
 - Determine how to add buttons and update fields
 - Determine how the user will interact with each particle
 - Should users be able to click an element of fluid and be able to adjust the force on it independently?
- Learn StepCore
 - Figure out how the World.doEvolve() function will interact with the Fluid object
 - Figure out how the solver will deal with fluid particles

Stage 2 – SPH Library (Week 3-4)

- Reduce the SPH Library from 3D to 2D
- Perform tests outside of Step to confirm fluid behavior is realistic
- Benchmark number of particles vs. frame rate

Stage 3 – Step Fluid (Week 5-8)

- Build the FluidParticle class
 - Make sure it is tested independently
- Build the Fluid class and FluidForce class
- Incorporate the Fluid objects into the Step GUI
 - This includes the graphics procedures to draw overlappable circles around each Fluid particle.

Step 3.5 – Interactions (Week 9) –

(For Midterm, Should be fluid testing in Step)

- Incorporate the Error Calculations into the FluidForce class
- Investigate the interaction of other objects in Step with fluid particles
- Is there realism?
- Do parameters need to be adjusted in order to replicate buoyancy and waterwheels?

Step 4 – Step Gas (Week 10)

- Research/Test how to either implement a gas using the Fluid classes or adjust the current Gas simulation to allow for a large number of particles
- Use the graphical interface of overlapping circles to draw a cloud-like representation of a gas (possibly more diffuse with transparency)

- This step includes testing not only the interatomic interactions of our fluid, but the interactions of the gas with other objects.

Stage 5 – Unforeseen Issues and Problem Solving Overhead (Week 11-12)

- This is a two week buffer for any unforeseen problems and inevitable bugs beyond my current understanding. These two weeks will surely be distributed at different points throughout the project.

Stage 6 – Documenting/Polishing (Week 13)

- Update Doxygen and any required documentation of design choices

Other Obligations

As of writing this proposal I have no obligations for that time period. However, I'm currently exploring possibilities for post-graduate research on some theoretical chemistry at my University this summer. I will also be improving my graduate school application by taking the Physics GRE test and taking sometime to let my undergraduate education sink in a bit. I should be able to contribute at least 20 hours per week on this project and I intend to take this project very seriously.

About Me

I'm a 4th year candidate for a Bachelor of Science at University of Waterloo in Ontario, Canada. My major of study is Computational Physics, which has equipped me with skills in programming, computational theory, and physics. Perhaps the most relevant courses I have taken in regards to this project would be computational differential equations, computational physics, and condensed matter physics. Although, I have never had a course specifically on fluid dynamics or continuum mechanics.

As far as coding experience I have been somewhat limited to small coding projects. Through a recent assignment in one of my courses, I had the chance to program a molecular dynamics simulation in python using a Leonard Jones potential in order to study its solid to liquid transition point as a function of temperature. However, most of my software development experience that should be relevant to this project has been through co-op work placements.

Most significantly, I worked for a summer on a real-time medical imaging application at Sunnybrook Hospital in Toronto, Ontario. Among a number of my responsibilities there, I had the pleasure to contribute to their existing C++ project, a properly documented and tested plugin. They had an existing Qt panel based GUI for their application and, using an existing plugin as a prototype, I wrote code which expanded the capability of their software to display multiple MR slices. Working on this Qt project was also very instructive on the nature of

signals and slots design principle.

I believe that this will be an excellent project for me to gain software development experience outside of strictly implementing numerical algorithms. I find the subject matter of computational fluid dynamics very interesting and have enjoyed investigating the peer-reviewed literature on the latest advancements in this field. I am looking forward to having an opportunity to contribute to an open-source project, something that I have never done before. Furthermore, the fact that this software is intended for educational use also makes it appealing to me as I feel that science literacy is extremely important for students of all ages.